Aggregation of Communication Data for Inner Source Software Development

BACHELOR THESIS

Philipp Wörndle

Submitted on 26 September 2023

Friedrich-Alexander-Universität Erlangen-Nürnberg
Faculty of Engineering, Department Computer Science
Professorship for Open Source Software

Supervisor:
Stefan Buchner
Prof. Dr. Dirk Riehle, M.B.A.

Friedrich-Alexander-Universität
Faculty of Engineering

Declaration of Originality

I confirm that the submitted thesis is original work and was written by me without further assistance. Appropriate credit has been given where reference has been made to the work of others. The thesis was not examined before, nor has it been published. The submitted electronic version of the thesis matches the printed version.

Erlangen, 26 September 2023

License

Erlangen, 26 September 2023

Abstract

Inner Source software development is becoming more popular and promises a lot of benefits to companies. Inner Source projects can involve distributed development teams, where communication management is challenging. There are tools like GrimoireLab for integrating and analyzing data of software projects. GrimoireLab only covers the integration of a small percentage of communication tools and their data. Even though there is research in analyzing communication data, there is a lack of data integration tools for enterprise communication tools. This thesis aims to integrate communication data from Cisco Webex and Microsoft Teams and provide a concept on how to aggregate this data to gain insights into Inner Source software development.

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
OSI	Open Source Initiative
ETL	Extract Transform Load
ELT	Extract Load Transform
SDK	Software Development Kit
JSON	Javascript Object Notation
OLAP	Online Analytical Processing

1 Introduction

Open Source software development describes specific practices in software development that aims on distributing the resulting source code publicly under an Open Source license (Open Source Initiative, 2000) This enables other people to access, understand, modify and enhance the code. The Open Source license regulates how a third party can use, copy, modify and distribute it. These Open Source principles set the foundation on which contributors get recognition and information can be accessed publicly.

Inner Source tries to bring Open Source principles to companies. This enables employees to work together in an Open Source manner and allows the flow of information as well as solutions in a company wide boundary, where the Inner Source software can be shared, used and collaborated on.

Inner Source software development can have a lot of advantages for companies, but adapting to these principles can come with challenges. Therefore it is important to analyze activities in Inner Source projects to gain valuable insights and prevent potential project drifts. This can be done by analyzing the meta data a project and its members produce. As a result, data driven decisions allow for easier adoption of Inner Source by enabling project managers to make better decisions. There already exists a lot of data analysis and integration on development data. But there is a lack of available communication data integration tools.

This thesis integrates communication data and proposes a concept on how to aggregate this data for effective Inner Source project analysis. In chapter 2 more insights into the integration, aggregation and analysis of communication data is given, showing the demand of integration tools using data pipelines. The requirements for this thesis are stated in chapter 3. Chapter 4 will propose the final architecture of the integration tool. Chapters 5 and 6 deal with the process of working off the requirements by designing and implementing a data integration tool. Afterwards the requirements will be evaluated in chapter 7. In the end chapter 8 will suggest extension topics on the presented integration tool for future research and work.

1.1 Software Development

Software development is a learning process in which knowledge is gained and information is generated throughout the projects lifetime. (Rubin and Wohlin, 2014) Effective project management builds upon successful team building, knowledge sharing, communication, and conflict resolution. Höst *et al.* classified software into four categories by analyzing the openness of the development process and its resulting product. Where traditional proprietary software represents a closed process and closed product, the opposite is true for Open Source Software, where anyone can participate in the development process and the resulting source code is publicly available. Höst *et al.* introduced a category called *Controlled Open Source Software* where the product offers an OSI-approved Open Source License, but the development process is strictly controlled by a single organization. Inner Source resembles the last category that imposes two main features - an open development process to all developers within an organization, while access to the resulting products is restricted to that organization. (Höst et al., 2014)

1.2 Inner Source

Inner Source tries to bring Open Source principles to organization wide boundaries. As Inner Source repositories are available within an organization, they give a good starting point for projects and can increase reuse.(Dinkelacker et al., 2002, Lindman et al., 2008, Vitharana et al., 2010) Open environments increase awareness (Lindman et al., 2008, Lindman et al., 2012) and innovation (Morgan et al., 2011, Lindman et al., 2012), and as a large developer pool is familiar with an organizations technologies (Dinkelacker et al., 2002, Riehle et al., 2009), it is easy for developers to switch teams or projects and therefore increase development speed.(Dinkelacker et al., 2002, Wesselius, 2008) In addition Inner Source Projects can benefit from Linus Law, which states *given enough eyeballs, all bugs are shallow*, which relates to improved product quality.(Dinkelacker et al., 2002, Riehle et al., 2009) Despite all benefits, the adoption of Inner Source can come with challenges. Capraro and Riehle propose a model consisting of eight adoption challenges. They categorized these challenges into two categories. (Capraro and Riehle, 2016) The first category of challenges are due to the mismatch of Inner Source practices and the traditional preexisting organizational setup. These challenges include resistance due to significant change, diversity causing a lack of collaboration among organizational units and local interests of organizational units,like the fear of resource loss or maintenance effort. The second category of challenges deal with Inner Source adoption itself. These include that it is difficult to utilize of openness, the application of control and steering, resentments against code transparency, the contribution process not running smoothly or not knowing what to Inner Source. (Capraro and Riehle, 2016) Judging from that, there seems to be a need for software analytics, monitoring the adapting process, helping companies transition to and benefiting from Inner Source.

1.3 Software Development Analytics

Software analytics can be used to help developers and project managers make better decisions by providing them with the insights they need to better understand their projects and how they are evolving. (Guerrouj et al., 2016) For example, software analytics can help developers and project managers identify code smells - areas of code that are prone to bugs - and track the progress of a project over time.Additionally, software analytics can help project managers to identify areas of improvement and potential risks by providing them with data-driven insights into the projects progress and performance. Furthermore, software analytics can be used to measure the effectiveness of a projects development process. (Guerrouj et al., 2016)

Integrating software analytics data from various sources has been identified as a critical role, yet it still presents a challenge due to the heterogeneity of data. (Martínez-Fernández et al., 2018) Different sources contain different types of information, and the same information is stored in different formats and tools. To tackle this issue, further research is needed to simplify useful analytics. Companies developing commercial systems also face the challenge of understanding how to combine heterogeneous data sources for software analytics to meet the true information needs of end users. (Martínez-Fernández et al., 2018)

2 Literature review

2.1 Communication Analysis in Software Development

In their book 'Software Project Management in a Changing World' Ruhe and Wohlin state that Software Project Management requires effective communication and coordination among team members to ensure successful completion of the project(Ruhe and Wohlin, 2014) Communication management involves establishing clear communication channels between stakeholders, team members, and project interests at different levels. Additionally, team building, knowledge sharing, and conflict resolution are important elements of successful project management(Ruhe and Wohlin, 2014) Inner Source orients itself on Open Source principles and often involves a distributed development team. Šmite states ten misconceptions in distributed software development. These misconceptions were researched by conducting empirical research in software companies from all over the world.One of these misconception is that "any problem can be fixed with the right toolset" (Šmite, 2014). She recognizes the importance of communication tools, though concludes that "... most problems in distributed development are human and not technical"(Šmite, 2014). She delegates the responsibility to the project manager(Šmite, 2014) Stamelos states that an Open Source projects success is dependent on keeping the contributors interested in the project. (Stamelos, 2014) Therefore it is important for decision making to know in which direction the project moves, whose contribution is crucial to the project and what drives their interest. This allows to plan future steps accordingly. There are two types of tooling, which derived from social media research - Social Network Analysis and sentiment or opinion detection tools. SNA provides analysis on important individuals from the project community, ranked by their contributions.Sentiment or opinion detection tools could allow measuring the agreement or disagreement of the community to specific topics (Stamelos, 2014) Sentiment and opinion detection tools rely on communication data.

2.2 Aggregation and Integration of Communication Data

There is a need for new tooling for data aggregation that can be used by project managers. (Buse and Zimmermann, 2010) Online Analytical Processing (OLAP) is used for aggregating and analyzing multidimensional data, typically numerical. OLAP provides operations for increasing and decreasing the level of detail when inspecting data. With the rise of text mining, the scope of OLAP has been extended to include textual data, which requires new aggregation functions. Text mining provides the necessary techniques to use OLAP with textual data, revealing a new research field. It is more challenging than traditional data mining and involves techniques from data mining, natural language processing, artificial intelligence, and machine learning. These techniques are used for information extraction, keyword aggregation, document categorization, and text summarizing. The key challenges of textual aggregation approaches are to reduce human efforts as much as possible while providing a high degree of accuracy and to process large volumes of data in a short amount of time. (Bouakkaz et al., 2017) Communication data can be categorized into meta data and content data. Meta data includes all information about the communication, except for the content. The content of a communication is textual information, whereas the meta data can consist of numerical values as well. Before data can be aggregated, it first needs to be integrated. Data pipelines are a collection of jobs that enable the automated integration of data from a source to a destination. (Van Alstyne et al., 2016) They involve a set of activities that manipulate data, with the output of one job becoming the input of the next. (Raj et al., 2020) Various types of data, such as continuous, intermittent, and batch data can be handled by data pipelines (Goodhope et al., 2012) They are broadly divided into two main categories: ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform). (Raj et al., 2020) The main difference between them being, where the actual transformation takes place. The transformation job could be before (ETL) or in the target system (ELT). Data pipelines lay the foundation for integration and aggregation of communication data for effective analysis to gain insights in Inner Source projects.

3 Requirements

The main goal of this thesis is to integrate different data from communication tools into one integration tool. Therefore an overview of relevant communication tools and the available data provided by these tools, should be created. In addition the thesis should propose a concept on how to unify and transform the communication data for innovative insight into communication behavior within Inner Source software development projects. If possible the created insight should be demonstrated on a real Inner Source project or simulated with mock data.

The integration of different communication data sources should be implemented with a given data pipeline framework. It was also agreed upon that python was picked as the programming language of choice, due to its popularity in the data analysis world and the already existing data pipelines. As there is already a tool named GrimoireLab that integrates a variety of different communication data sources, it was required to choose data sources that haven't been covered by GrimoireLabs yet.

The project was developed in an agile manner. In weekly meetings the progress was discussed and the requirements were adjusted to new findings, opportunities and limitations.

4 Architecture

The chosen architecture accomplishes two main goals - implement data pipelines that build own modules that can be used in a plug and play manner and create a clear separation of concerns making it easy to extend applications with new data sources.

4.1 Current Data Pipelines

The current data pipelines are implemented in python and it was mandatory to use them.They follow an approach similar to the ETL pipelines discussed previously and are implemented using the bridge pattern.

Bridge Pattern

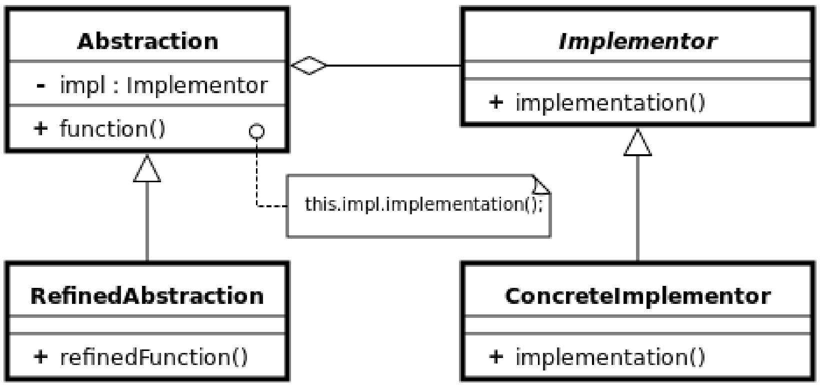


Figure 4.1: Bridge Pattern Class Diagram

The structural design pattern bridge decouples an abstraction from its implementation in a way that allows independent variation of both. (Gamma et al., 1995) The purpose of the bridge pattern is therefore to create a flexible and expandable architecture that allows for adding new functionality without changing existing code. It is useful when there is a hierarchy of classes of different implementations that share one abstraction. This could be used e.g. to change implementation of classes at runtime or separate tightly coupled logic. (Gamma et al., 1995)

Pipeline Structure

The general structure of the bridge pattern can be adapted to fit the requirement that proposes universal data pipelines, by dividing a single pipeline logic in its most basic building blocks that can be replaced in the future to fit unique ways to request, transform and write the data. A single *Pipeline* holds a collection of *PipelineStep*s that can be executed. A *PipelineStep* is a wrapper around an abstraction. The given abstractions were *Requester*, *Transformer* and *Writer*. This allows building abstract pipeline logic independent of concrete data source implementations, changing the way of access to data sources at runtime and decoupling the system's storage from the pipeline logic.

4.2 Communication Data Pipelines

The goal is to provide a layered architecture that separates the concerns that occur when extracting, transforming and loading data from a source to a target. Every layer builds upon the last and serves its output to the next layer as input. Therefore an application that is written once, does not need to be adjusted when plugged with data from a different data source. Every layer consists of specific modules managing individual data pipeline. A domain represents a collection of metrics and unifications that are specific to a certain topic e.g. communication or development. Even though some unifications might be shared with other domains, every domain has their unique metrics.

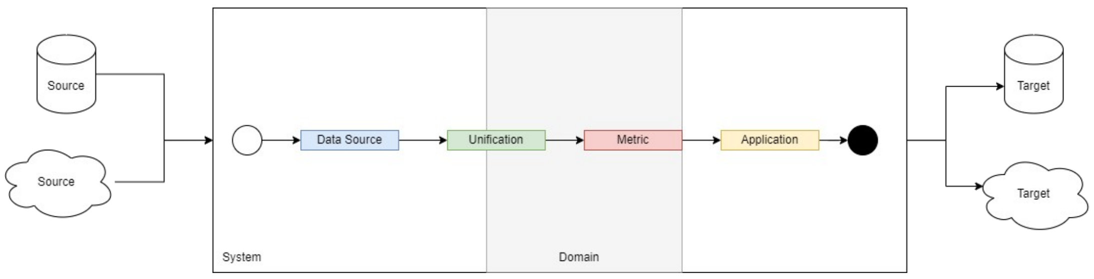


Figure 4.2: System Architecture

Data Source Layer

The data source modules enable their user to retrieve models without any knowledge on the Application Programming Interface (API) specifics. This includes constraints or dependencies to other endpoints and how to interact with them. There will be a module for every data source managing the individual data source pipeline for each endpoint. They provide an entry point for raw data into the system. The data source module bundles the logic to interact with a remote API, resolve model dependencies and the configuration of the data source pipelines.

Unification Layer

When integrating data from different sources, they often provide conceptually similar information that uses different semantics. They also provide a variety of different attributes in various data formats. The unification modules provide access to unified data wrapping around a data source module. Unification pipelines serve as a data gateway, where raw data is processed to produce system or domain entities. These entities provide a universal naming convention, format structure and can support relational modelling. A unification module will be responsible for generating unified data by leveraging access to a data source module and configuring unification pipelines.

Metric Layer

After the data is unified we can enrich the data with domain specific metrics or define custom relations. These can be modelled independent of specific data sources using unified entities as an interface. Entities on a concrete level will be represented by python *dataclasses* that allow for defining type safe classes. A metric module consists of different ways to aggregate data by configuring different metric pipelines. This offers the possibility to share domain specific metrics among applications.

Application Layer

When gaining insights from data, applications encapsulate specific logic that can model a given usecase. These usecases can have dependencies to multiple different domains and therefore need to resolve these logic dependencies optimized to their application needs. Application pipelines therefore set up and connect the modules from the layers below and determine the applications individual system output. The application modules allow for the creation of different application stages e.g. develop and production, by configuring different application pipelines.

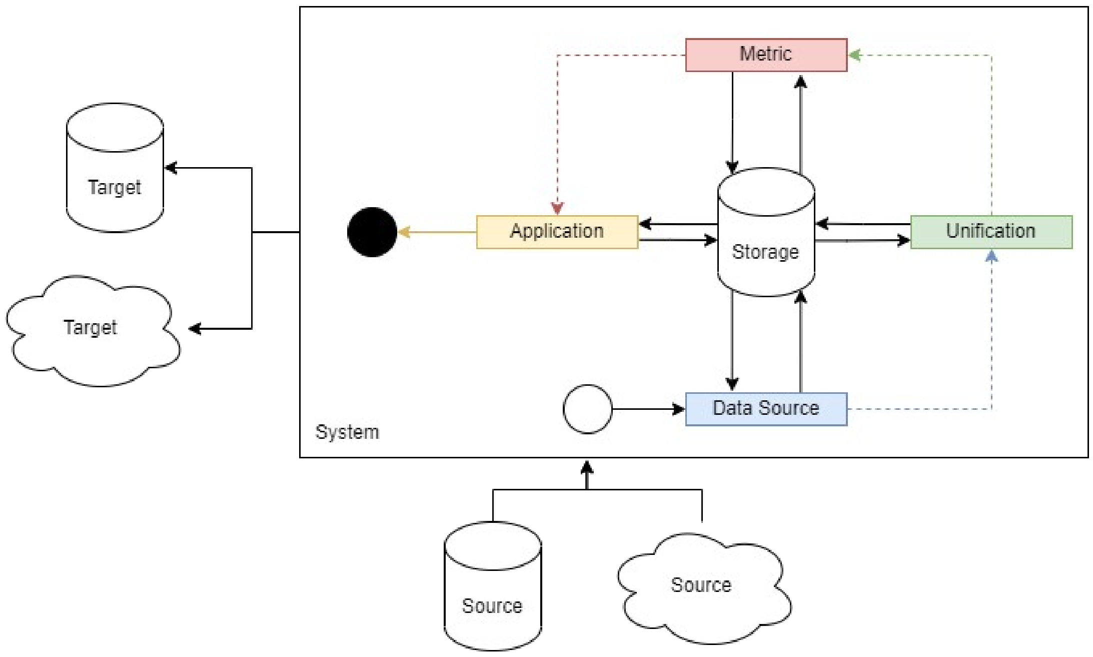


Figure 4.3 System Storage

System Storage

The system should offer the possibility to persist data. In the picture the straight-line arrows show the actual data flow, where every data pipeline interacts with the same storage. The dashed-lined arrows symbol the semantic data flow through the system. For this thesis, the chosen data storage was the local file system using Javascript Object Notation (JSON) documents. JSON is a human readable data format that therefore allowed for easy data inspection and fast development, as no data schema had to be provided.

5 Design

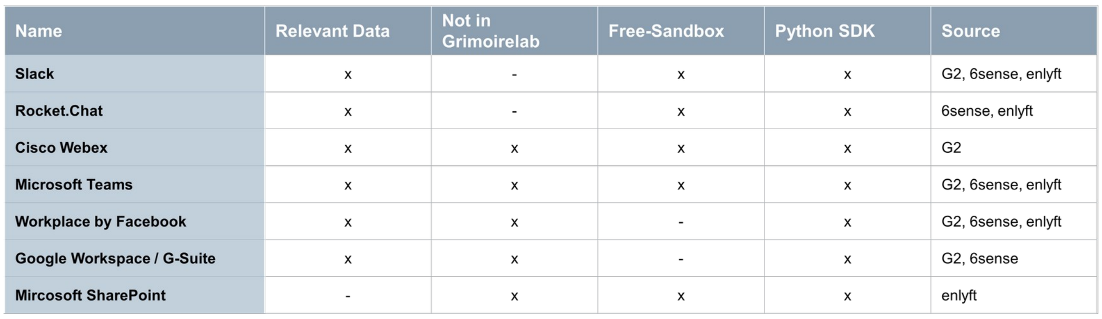
This chapter deals with the research in relevant communication tools and their available communication data. It also showcases the workflow of a data pipeline application by designing a demo application. GrimoireLab already offers the integration of communication data for Slack and Rocket Chat. These include data about a *User*, a communication *Channel* and the *Messages* posted by a user in a communication channel. This served as the baseline for further comparison and decision making.

5.1 Available Communication Tools

To find relevant communication tools market share research was done, by looking at G2, enlyft and 6sense. G2 provides review data for *Business Instant Messaging Software*. These can be filtered for company size. As Inner Source is more common in larger organizations the *Enterprise* option was selected, where *Enterprise* meaning companies having 1001+ employees. The communication products are ranked on satisfaction and market presence. The outliers in the category *Business Instant Messaging* in high performance and market leaders where Microsoft Teams, Cisco Webex and Slack. Enlyft offers market share data for *Collaborative Software*. According to their data, the majority of market share is held by Microsoft SharePoint, Slack and Workplace by Facebook. Enlyft does not provide insights into the company size using the product. 6sense provides market share data for *Instant Messaging and Chat Technologies*. According to 6sense, the market leaders are G Suite (Google Workspace) and Slack. 6sense also does not provide insights on the individual company sizes.

5.2 Data Source Evaluation

The mentioned data sources were evaluated by the following criteria: Does the tool provide relevant communication data? Is the tool not yet supported by GrimoireLab? Is there a free sandbox environment offered? Is there already a python SDK available? The following table concludes the results of this evaluation. Slack and Rocket Chat are already supported by GrimoireLab. Workplace



Name	Relevant Data	Not in Grimoirelab	Free-Sandbox	Python SDK	Source
Slack	x	-	x	x	G2, 6sense, enlyft
Rocket.Chat	x	-	x	x	6sense, enlyft
Cisco Webex	x	x	x	x	G2
Microsoft Teams	x	x	x	x	G2, 6sense, enlyft
Workplace by Facebook	x	x	-	x	G2, 6sense, enlyft
Google Workspace / G-Suite	x	x	-	x	G2, 6sense
Mircosoft SharePoint	-	x	x	x	enlyft

Figure 5.1: Communication Tools Evaluation

by Facebook and Google Workspace do not offer a free sandbox environment for testing. Microsoft SharePoint does not offer relevant communication data for this thesis. The choice was made in coordination with the supervisor to go with integration of Cisco Webex and Microsoft Teams and providing the possibility to inject data integrated by GrimoireLab for Slack and Rocket Chat.

5.3 Available Communication Data

Cisco Webex offers two main features relevant to communication - meetings and messaging. These are enriched with meta data about communication structure, like information about teams and their rooms, where users can interact.

Microsoft Teams also provides meeting and messaging features, but data is provic by the Microsoft Graph API. This gives access to a variety of additional communication data like information about calendars, mails or files.

5.4 Design Demo Application

To demonstrate the proposed architecture and data integration, a demo application should be created, giving insights into an Inner Source project by analyzing communication data. Keyword aggregation is a possible way to aggregate textual data (see 2.2). The chosen application will evaluate which keywords were used most often in specific channels and find the user that wrote the word most often. The unified data needed would include *Users*, *Channels* and *Messages*. A metric would be the aggregation of all words and users in a specific channel. The application layer would be responsible for evaluating the provided communication metric, by counting the absolute occurrences and adjusting the data structure to enable visualization.

6 Implementation

This chapter deals with the integration of the communication tools Cisco Webex and Microsoft Teams. It also provides insight into the injection of data integrated by an external tool like GrimoireLab. Afterwards the implementation of the previously designed demo application is shown. The pipeline was implemented to support batch data and therefore tries to request all available data, then applies transformation and afterwards write all data to storage at once.

6.1 Integrate Communication Tools

To integrate a communication tool into the system a data source module, managing different data source pipelines for each endpoint, needs to be created. To make use of the data on an metric and application layer, there also needs to be a unification module, managing different unification data pipelines.

Implement Data Source Pipelines

A single data source pipeline handles two types of different input data. Depending on the input data a corresponding *Requestor* is chosen. If a file pointer in form of a file name is provided a *JSONReader* requests the input from a local file. If no file pointer is provided the default input will be provided by an API client, which will fetch the data from the corresponding web API with the option to provide additional query parameters, to filter or limit specific data. If there are any model dependencies, they are resolved prior to the pipeline execution. E.g. in order to retrieve messages an ID of a specific user needs to be provided. This resolution is done by executing another data source pipeline in the same module. In both cases the data is then passed on to the data source transformer. A *DataSourceTransformer* will perform the following three steps. First filter instances based on a provided condition, which allows for coarse grained data cleaning. After that individual attributes can be white- or blacklisted. The last step offers the possibility to apply a specific transformation to each object, to reduce data size that gets passed around and was mainly used to resolve model dependencies.

In the end the data is returned in-memory by a *Writer* by default, offering the possibility to also persist it in storage. Modules for Cisco Webex and Mircosoft Teams were implemented against client Software Development Kit (SDK)s. In consultation with the supervisor it was decided to exclude integration of meeting data and mirror communication data available from the GrimoireLab integrations of Slack and Rocket Chat - *Users*, *Channels* and *Messages*. The module for Cisco Webex gives access to *Teams*, *Team Memberships*, *Rooms*, *Room Memberships*, *Messages*, *Peoples*, *Organizations* and *Roles*. The module for Microsoft Teams allows the integration of *Teams*, *Team Members*, *Channels*, *Channel Members*, *Messages*, *Users* and *Mail*. The communication tools data is integrated using the tools API semantic. The data source modules provide raw data in form of python dicts and therefore produces not type safe data. This makes it easy to handle different types of responses from the API, as there are cases, where not all attributes are set on a specific object and therefore do not show up in the response. It also reduces maintenance effort, because changes in the API response do not need to be copied on this layer, passing the responsibility for default values up to the unification layer.

Implement Data Unification Pipelines

For each communication tool an additional unification module is created. A unification module is built upon a data source module. The purpose of the unification module is to retrieve unified data without any knowledge of the underlying data source. Therefore the default functionality is accessing raw data from its corresponding data source module. But the unification module also provide a way to inject raw data in memory or via a file pointer. Depending on the provided input data a *Requestor* is chosen and the respective unification pipeline is triggered. A unification pipeline is then responsible for transforming the raw data according to a specific mapping. In most cases the mapping consists of renaming attributes, providing default values and transformations in data types and data formats. Especially when it comes to date and time a universal format should be chosen.

Integrate GrimoireLab Data

To inject externally integrated data from a tool like GrimoireLab a unification module needs to be created. This unification module is then responsible to unify the data according to the schema provided by GrimoireLab. Unification Modules were created for GrimoireLab Slack and Rocket Chat, offering *Users*, *Channels* and *Messages*.

6.2 Implement Demo Application

The demo application needs unified *Users*, *Channels* and *Messages*. The proposed metric should aggregate words on a *Channel* level. The application logic is then evaluating this metric.

Implement Metric Pipeline

In the demo application the metric pipeline resolves the relations of individual *Channels*, *Users* and their *Messages*. It aggregates them, so that *Messages* get split into individual *Words* and structures them. Each *Channel* holds a list of available *Words* and each *Word* holds reference to the *Messages* in which it was used and when the *Message* was posted and by whom. The metric pipeline is used by the communication metric module, which in future will manage different metric pipelines regarding the calculation of communication metrics. Such a metric could include the enrichment of the *Messages* with a sentiment or provide special types of data cleaning in the context of communication data, e.g. remove all punctuation marks.

Implement Pipeline Application

On the application layer specific system logic can be encapsulated. The application module configures a single application pipeline. Therefore it manages where the data request its configuration - authentication information - and where to write or publish the resulting data. This can differ for different application stages like development or production. The request part consists of configuring and connecting of modules from lower layers, to compute all necessary metrics for the individual application. In the configuration part we need to provide authentication information for the chosen data source module. For Webex this could either be an *access token* or *client id* and *client secret*. The Microsoft Graph API needs in addition to *client id* and *client secret*, also a *tenant id*. These need to be set up in advance and given the application via command line argument or environment variable. The system could be monitored by providing each module and pipeline a global logger. Each data source module gets injected to the corresponding unification module. In addition the metric modules are set up and the unified data is requested. The *Users*, *Channels* and *Messages* are then passed on to calculate the *ChannelWordFrequency* metric. The resulting metric is passed on to the *Transformer* where the metric is evaluated. The result is then either persisted in storage and or returned in memory for visualization.

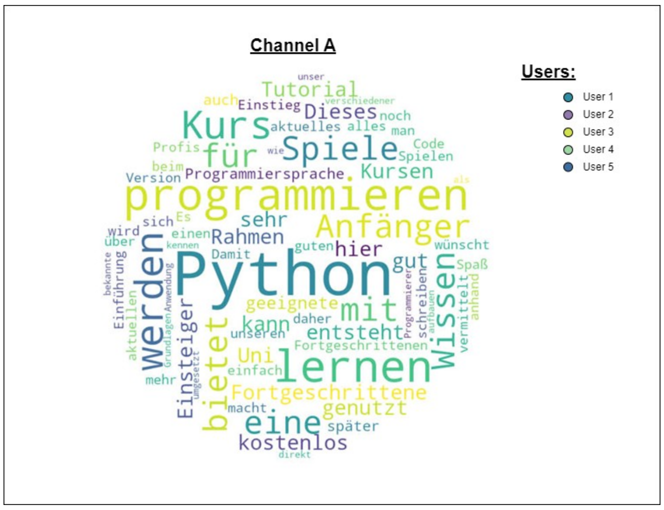


Figure 6.1.Keyword Analyzer Visualization Concept

Visualization Concept

As no representative data was found on Cisco Webex and Microsoft Teams, mock data was used and injected on the unification layer to demonstrate the application resultsA proposed visualization concept would include a *wordcloud* per channel.The word size encodes the absolute occurrences of the word in the specific channel.The word color would encode the user who wrote that word most often.

7 Evaluation

An overview of relevant communication tools was created. The available market share data was limited. As the data from G2 consists of reviews, they might underlie bias. The category provided by enlyft called *Collaborative Software*, differs from G2 and 6sense, which offered data about *Instant Messaging*. The provided data quality is difficult to compare.

The proposed concept to integrate and transform the data to derive insights consists of four layers. These layers create a separation of concerns to write an application once and extend it with new data sources. It also enables sharing of modules and reuse of data pipeline logic. New communication tools can be integrated to the system by creating a data source module. To use their data in metric calculations an unification module needs to be created additionally. The proposed architecture allows extensions on three levels - integrate new communication tools, model domain metrics or build custom data applications. The created modules on each layer provide the necessary functionality to plug-and-play, while the data pipelines denote and enforce the common functionalities on each layer. Therefore an application that is written once, can easily be extended to include new data sources, new entities or new metrics from possibly different domains. Good documentation is provided, as this adds another layer of abstraction that can make it challenging for beginners to start building modules and pipelines.

The integration of communication data was done for Cisco Webex and Microsoft Teams. Most of the available communication data available was integrated for Cisco Webex except for meetings. For Mircosoft Teams there is more data that could be relevant like meeting artifacts or calendar events. The injection of externally integrated data was done for GrimoireLabs integration of Slack and Rocket Chat data. Demonstration was done with mock data, as neither Open Source nor Inner Source projects were found using the data sources or be willed to provide their data.

8 Conclusions

An overview of relevant communication tools was given. There seems to be a lack of accurate available data on what communication tools are used by Inner Source projects in particular.

The provided integration of relevant communication data was implemented for Cisco Webex and Microsoft Teams. Data integrated by GrimoireLab for Slack and Rocket Chat can be used as well. Even though a lot of relevant communication data was integrated, there is even more data provided by these communication tools. The provided pipeline framework was used.

The proposed concept for a data pipeline application was demonstrated with mock data. Future work could build upon this tool and either extend existing data sources, integrate more data, model new domains or applications, or extend the proposed architecture. This concludes that all the requirements have all been met.

For integrating more data sources research could be done surveying Inner Source projects, on what tools the project uses. Other relevant domains could include developer data or data about the organizational structure. Extending the proposed architecture could include support for intermittent data, quality gates for monitoring purposes or replacing the file system with a database.

In summary communication data was integrated and aggregated to gain insight into Inner Source projects. This lays the foundation for effective software analysis tools using this data integration tool to access communication data.

References

Open Source Initiative. (2000). The Open Source Definition [Online; accessed 26 August 2023].

Bouakkaz, M., Ouinten, Y., Loudcher, S., & Strekalova, Y. (2017). Textual aggregation approaches in olap context: A survey. *International Journal of Information Management*, *37* (6), 684–692.

Buse, R. P., & Zimmermann, T. (2010). Analytics for software development. *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 77–80.

Capraro, M., & Riehle, D. (2016). Inner source definition, benefits, and challenges. *ACM Computing Surveys (CSUR)*, *49* (4), 1–36.

Dinkelacker, J., Garg, P. K., Miller, R., & Nelson, D. (2002). Progressive open source. *Proceedings of the 24th International Conference on Software Engineering*, 177–184.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Pearson Deutschland GmbH.

Goodhope, K., Koshy, J., Kreps, J., Narkhede, N., Park, R., Rao, J., & Ye, V. Y. (2012). Building linkedin’s real-time activity data pipeline. *IEEE Data Eng. Bull.*, *35* (2), 33–45.

Guerrouj, L., Baysal, O., Lo, D., & Khomh, F. (2016). Software analytics: Challenges and opportunities. *Proceedings of the 38th International Conference on Software Engineering Companion*, 902–903.

Höst, M., Stol, K.-J., & Oručević-Alagić, A. (2014). Inner source project management. *Software Project Management in a Changing World*, 343–369.

Lindman, J., Riepula, M., Rossi, M., & Marttiin, P. (2012). Open source technology in intra-organisational software development—private markets or local libraries. In *Managing open innovation technologies* (pp. 107–121). Springer.

Lindman, J., Rossi, M., & Marttiin, P. (2008). Applying open source development practices inside a company. *IFIP International Conference on Open Source Systems*, 381–387.

Martínez-Fernández, S., Jovanovic, P., Franch, X., & Jedlitschka, A. (2018). Towards automated data integration in software analytics. *Proceedings of the*

International Workshop on Real-Time Business Intelligence and Analytics, 1–5.

Morgan, L., Feller, J., & Finnegan, P. (2011). Exploring inner source as a form of intra-organisational open innovation.

Raj, A., Bosch, J., Olsson, H. H., & Wang, T. J. (2020). Modelling data pipelines. *2020 46th Euromicro conference on software engineering and advanced applications (SEAA)*, 13–20.

Riehle, D., Ellenberger, J., Menahem, T., Mikhailovski, B., Natchetoi, Y., Naveh, B., & Odenwald, T. (2009). Open collaboration within corporations using software forges. *IEEE software*, *26* (2), 52–58.

Ruhe, G., & Wohlin, C. (2014). *Software project management in a changing world* (Vol. 21). Springer.

Šmite, D. (2014). Distributed project management: Ten misconceptions that might kill your distributed project. In *Software project management in a changing world* (pp. 301–320). Springer.

Stamelos, I. (2014). Management and coordination of free/open source projects. In *Software project management in a changing world* (pp. 321–341). Springe

Van Alstyne, M. W., Parker, G. G., & Choudary, S. P. (2016). Pipelines, platforms, and the new rules of strategy. *Harvard business review*, *94* (4), 54–62.

Vitharana, P., King, J., & Chapman, H. S. (2010). Impact of internal open source development on reuse: Participatory reuse in action. *Journal of Management Information Systems*, *27* (2), 277–304.

Wesselius, J. (2008). The bazaar inside the cathedral: Business models for internal markets. *IEEE software*, *25* (3), 60–66.